Regression Analysis of Open Baltimore's Calls for Service Dataset

By:
Jerson Guansing, Daniel Jun and Tae Song
CMSC 455: Numerical Computations
University of Maryland, Baltimore County (UMBC)
Baltimore, MD 21250

Abstract

The purpose of this project is to utilize a pre-existing, massive database in order to effectively predict an outcome given a set of data. For this project we will be utilizing the Open Baltimore database, specifically 911 emergency calls, to show how our method may be used to predict the volume of calls given certain parameters such as time of the day, the day of the week, or different months. Using a least square fit method, a multivariable regression model will be created with the variables representing several data provided by the database and the output representing the volume of calls.

1. Introduction

This report analyzes the 911 calls, call distribution, and how each column of data from Open Baltimore dataset factors in the total call volume for any given date, district, etc. The size of the dataset (over 2 million entries) is what initially attracted us to this topic. Given recent development regarding crime and police work in Baltimore, we thought analyzing 911 calls would be a great way to learn a little more about our city.

Analyzing all 911 calls in the past two years will allow us to make good predictions on the busiest times of the year, month, and day for the 911 call centers. The information we find could be used by the Baltimore County for a variety of scheduling purposes. For example, Baltimore County could assign more police officers to work during times with the highest volume of calls dealing with crime, spam the public with fire safety announcements during times with the highest fire department calls, and schedule general technical maintenance during times with the lowest volume of calls. Predicting the volume and type of calls at specific times would be invaluable in saving taxpayer dollars by maximizing the efficiency with which the state of Maryland appropriates funds for emergency services. Furthermore, if done correctly, the same method could be applied to every other state in the United States.

2. Background

The main goal of Baltimore's 911 Communication Center is to receive calls regarding potential emergencies, and dispatch the appropriate personnel as swiftly as possible. They are the first line in responding to a 911 emergency as they are tasked with the critically important and difficult job of speaking to panicked citizens in life threatening situations, garnering the most pertinent information from them, and contacting the appropriate emergency personnel to the correct locations.

Though Baltimore's 911 Communication Center receives calls from every district in the county and city, Baltimore City is the busiest in the entire state, according to the Baltimore Sun as they handle over 4,000 calls a day (Prudente, 2016). The 911 center is operational 24 hours a day, 365 days a year and provides the vital first step in handling calls from citizens for ambulance, police and fire department assistance. The logistical requirements are very high in order to keep the life saving service operational in terms of resource and manpower allocation.

3. Methodology

3.1 Languages Used

The project is written in Python using three scripts. The first one, getAndCleanData.py, downloads the dataset from Open Baltimore's website and processes it by cleaning, removing outliers and generating an output file that will be used for the data distribution and regression analysis. The second file, Main.py, imports the generated file and creates a model that will be used by the polynomial regression. It will prompt for an input, and then calculate the estimated call volume given the user criteria. The third file, Counters.py, is used as a backward port for versions of Python that are missing the collections library.

Additionally, a Jupyter notebook (Python) was used for data distribution analysis and to generate graphs.

3.2 Machines

We used our personal machines to develop the python source code to download, manipulate, and conduct statistical analysis on the data. We chose Python because it is a quick and simple language to write, compile and run, and because of the extremely useful Numpy library which contains many statistical tools. We ran the three scripts on UMBC's gl system because we are all familiar with the gl system and gl was sufficiently quick enough. The Jupyter notebook requires additional libraries that are not available on the GL server such as pandas and matplotlib.

3.4 Inputs and Testing

We obtained the dataset from the Open Baltimore website which contains a plethora of data about Baltimore. The dataset, Calls for Service (911 calls), can be downloaded as a CSV or JSON format. This project downloads the CSV format which is over 200 mb in size or over two million call entries.

The downloaded dataset had the following columns:
callDateTime, priority, district, description, callNumber, incidentLocation, location

The data is then cleaned and manipulated to be consistent with the rest of the data (removing entries that are missing crucial data, splitting columns, etc.). Also, outliers (data distribution is less than 0.1% of the total) have been removed.

The column callDateTime have been split into 5 columns: month, day, year, callHour (converted to 24 hour format) and callMin. Additionally, the columns dayOfWeek and dayOfYear have been inferred by passing the call date (mm/dd/yyyy)   to a corresponding Python function. The columns priority and district have been converted to numeric values by pulling out unique entries as unique keys for each respective columns. The column location have been split to longitude and latitude columns where the default value is 0 for records that are missing the entry.

The columns description, callNumber (unique id for each entry) and incidentLocation have been removed due to high unique key entries and lack of discernable standardized values (thousands of unique keys).

To develop and test models, we will use a few curve fitting techniques.

Least Square Fitting

For this project, the Y value is the call volume, and X1,X2,...,Xn are the columns in the dataset pertaining to each call record. Each entry in the X array will be a vector of features of the data [X1,X2…,Xn] such as date, time, and district. The array of Y values will represent the actual call volume given a feature of the data. The features included the day, month, year, day of the year, day of the week, priority, district, longitude, and latitude of the emergency call as well as the duration in hours and in minutes of the call. We are testing two least squares fitting: linear and polynomial

Least Square Fitting - Linear
In order to perform the Least Square Fitting on the data sets, we find a, b and c such that
Y_approximate = y0 + a * X1 + b * X2 + c * X3 + … + coefficient*Xn
The method for determining the coefficients directly followed the form given below.

```
| SUM( 1* 1)   SUM( 1*X1)   SUM( 1*X2)   SUM( 1*X3) |     | y0 |   | SUM( 1*Y) |
| SUM(X1* 1)  SUM(X1*X1)  SUM(X1*X2)  SUM(X1*X3) |     | a  |    | SUM(X1*Y) |
| SUM(X2* 1)  SUM(X2*X1)  SUM(X2*X2)  SUM(X2*X3) | x | b  | = | SUM(X2*Y) |
| SUM(X3* 1)  SUM(X3*X1)  SUM(X3*X2)  SUM(X3*X3) |     | c  |    | SUM(X3*Y) |
```

Least Square Fitting - Polynomial
In order to perform the Least Square Fitting on the data sets, we find a0 to ak such that
Y_approximate = a0 + a1*x + a2*x^2 + … + ak*x^k
The method for determining the coefficients directly followed the form given below.

```
| SUM(1  *1)    SUM(1  *Xi)    SUM(1  *Xi^2)    SUM(1  *Xi^3) |        | C0 |    | SUM(1  *Y) |
| SUM(Xi  *1)   SUM(Xi  *Xi)  SUM(Xi  *Xi^2)   SUM(Xi  *Xi^3) |        | C1 |    | SUM(Xi  *Y) |
| SUM(Xi^2*1) SUM(Xi^2*Xi) SUM(Xi^2*Xi^2) SUM(Xi^2*Xi^3)|  x   | C2 | = | SUM(Xi^2*Y) |
| SUM(Xi^3*1) SUM(Xi^3*Xi) SUM(Xi^3*Xi^2) SUM(Xi^3*Xi^3) |      | C3 |    | SUM(Xi^3*Y) |
```

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^k \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^k & \sum_{i=1}^{n} x_i^{k+1} & \cdots & \sum_{i=1}^{n} x_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^k y_i \end{bmatrix}.$$

Solving Linear Equations
A Numpy Python library (Numpy.linalg.solve) was used in place of a Gaussian Elimination function to solve for the coefficients.

Polynomial Regression
The polynomial coefficients returned by the numpy.linalg.solve function was used to create a regression model to calculate approximate y values. The accuracy of the function was measured by calculating the following errors:

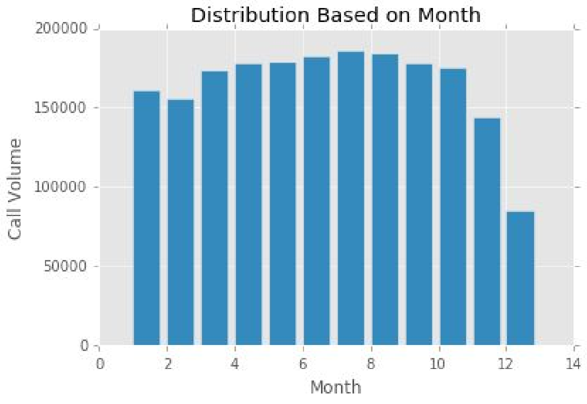
Least Square Fitting - Linear
Y_approximate = intercept + a*x1 + b*x2 + c*X3 + … + coefficient*Xi

Least Square Fitting - Polynomial
Y_approximate = intercept + a*x1 + b*x2^2 + … + coefficient*xi^k

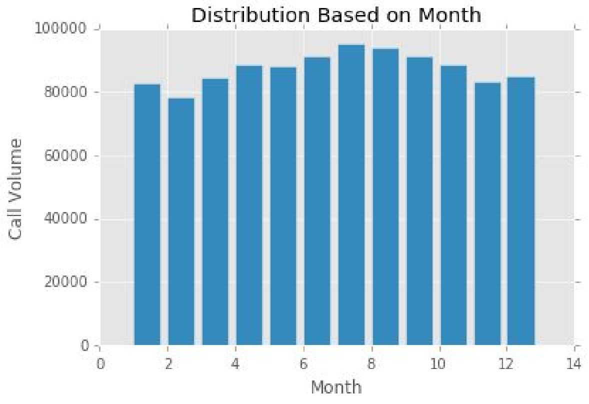
Absolute Error = abs(Y_actual - Y_approximate)
Max Error = max(Absolute Error array)
Average Error = Sum(Absolute Error) / number_in_set
RMS Error = sqrt (sum_over_set(Absolute Error ^2) / number_in_set)

3.4 Data Distribution Analysis

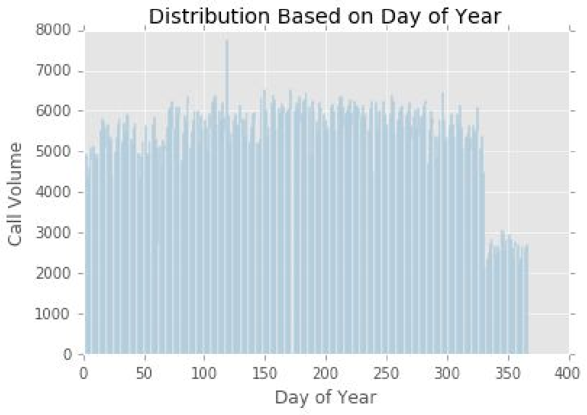
Analyzing the data distribution is necessary to visualize patterns and determine possible outliers in the data. The graphs below have already been cleaned where outliers (less than 0.1% of the total) were removed.
Call distribution based on the month and day of the year (365 or 366 days on leap years).



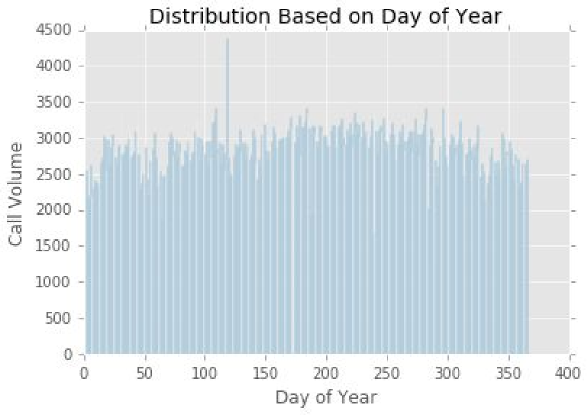
2015 - Present
Call volume for December is lower



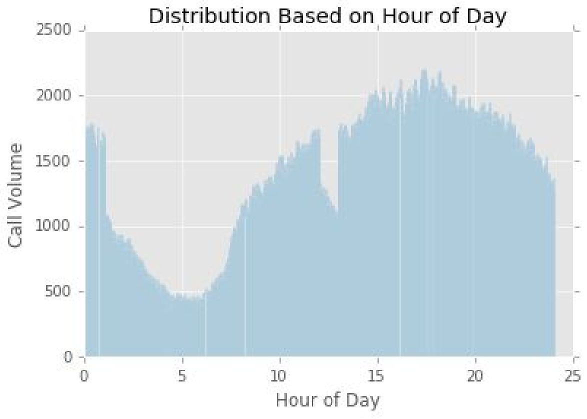
2015 - Only

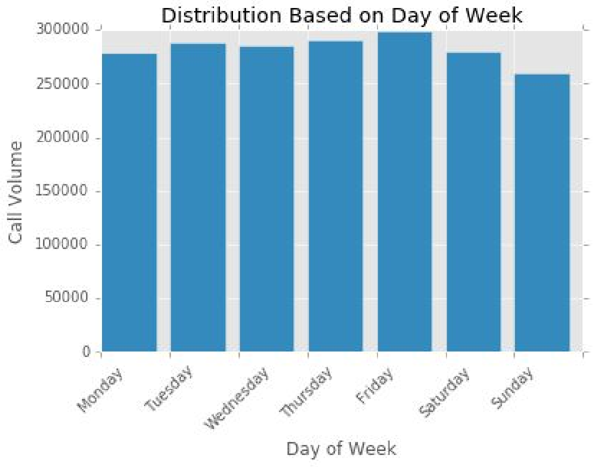


2015 - Present
Call volume for December is lower

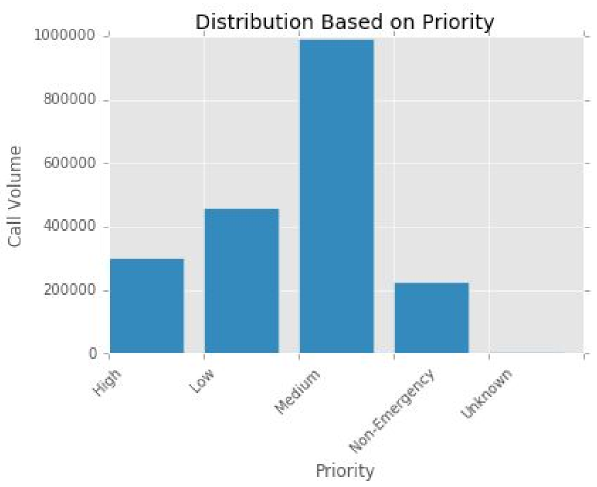
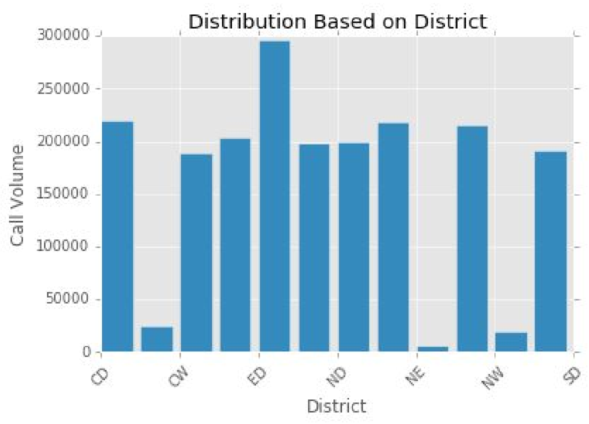


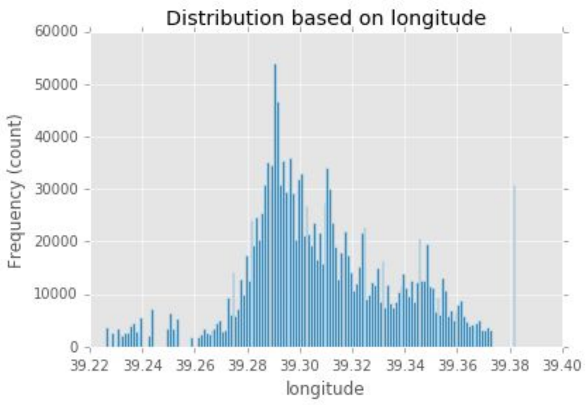
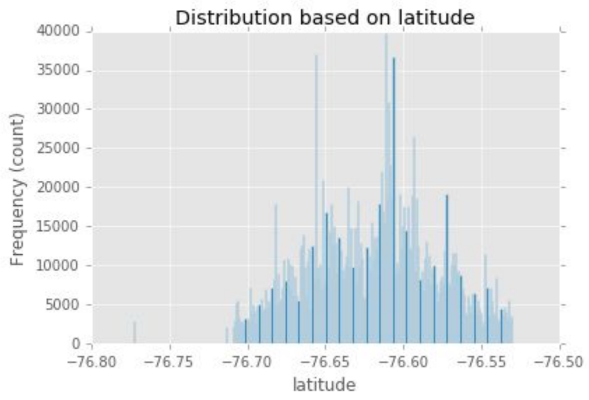
2015 Only





The call volume starts to increase starting at 2:00PM and peaks around 5:00PM. This is a good criteria to include because the data distribution doesn't just plateau (uneven distribution).	Call volume is highest on Fridays and lowest on Sundays.

 The majority of the calls are medium priority. Gain, this is a good criteria to include because there is variability in the data distribution.	 This is another feature that can be incorporated into our model because of the variability in the data distribution.

4. Implementation Results

To accurately predict and present call volumes, we modeled the data using curve fitting and polynomial regression.

4.1 Curve Fitting

We used Python to apply a variety of fitting techniques to the data. Least Square Fit and Polynomial FIt.

Coefficient Table based on N Degree

Refer to the train test evaluation for an overlapping comparison models for each N.

N	Fit	Intercept	month	day	year	dayOfYear	callHour	callMin	dayOfWeek	priority	district	longitude	latitude
11	Linear	2.98487830628	-0.039759213955 4	-0.00128030566679	-0.000956872342993	0.0013011285784 9	8.50207803995e-05	5.15137869747e-06	-2.29640693153e-05	-0.000690167143709	-0.000166587658681	-0.00134206190456	-0.000622966489293
11	Poly	2.01811051756e-33	3.64849842569e-29	8.24514116058e-31	1.64926158796e-23	2.17113340214e-23	1.91186582964e-27	1.98236979976e-23	1.23478913205e-28	8.95157367486e-29	-9.77586392028e-23	4.58429810212e-17	-1.13472106797e-21
10	Linear	2.21559681609	-0.036182591113 7	-0.00116467474975	-0.000576995014914	0.0011848873139 9	0.000122145507725	5.94561945727e-06	-4.25173270319e-05	-0.000971633974234	-0.000164865975714	-8.12637803689e-05	
10	Poly	1.51632639816e-20	8.2175001984e-20	4.673454861e-18	1.24125534165e-10	-2.06380864277e-14	2.30147591235e-14	1.57144423476e-14	7.5121218673e-16	2.01396633271e-17	-1.4882686675e-13	-4.72227537275e-19	
9	Linear	-2.53758361349	-0.010211063354 7	-0.000266605899122	0.00177733513401	0.000345242269006	0.0012134949221	1.42041481543e-06	-0.000553530113632	0.00679067389869	-0.000810357935902		
9	Poly	4.15626571814e-14	1.3410159648e-08	2.59630377009e-06	1.29241524369e-10	-1.3179608569e-13	1.98892053741e-09	2.10190019822e-14	-1.56456664743e-08	-1.49378478385e-06	-2.98551452791e-12		
8	Linear	-6.79551095605	0.198310729628	0.00677660842859	0.00384936718441	-0.00640415896456	0.0118060995401	-8.89316199606e-06	-0.0068662174535	0.0708707640242			
8	Poly	1.03250469245e-13	9.74670347081e-10	1.11715423358e-05	1.74660455999e-10	-1.29321802144e-12	1.82627461534e-08	1.86717366569e-13	-2.02593123436e-07	-1.4059666597e-05			
7	Linear	-15.1594276483	0.847876009162	0.0283230267176	0.00802036030774	-0.027673159812 2	0.0509365704493	8.90098681446e-05	-0.0212641254091				
7	Poly	4.9256425104e-13	-2.97060231732e-08	3.11537455028e-05	2.90726788342e-10	-4.86022622264e-12	9.86965974595e-08	6.08227860092e-13	-7.66944132423e-07				
6	Linear	-14.5830993	0.838613639499	0.0279935522785	0.00770661169595	-0.027365068386	0.0510247763768	8.8810507405e-05					
6	Poly	4.61618492369e-13	2.13459120562e-07	2.95035424907e-05	2.85749643457e-10	-4.78454028558e-12	9.88007315372e-08	6.05364261725e-13					
5	Linear	-3688.12712045	65.9956017043	2.19229942818	1.82770634747	-2.14854944714	4.68107233784						

5	Poly	2.7661405892e-11	-1.65439660901e-06	0.00208442646049	1.31231997491e-08	-2.74338543768e-10	1.05048181001e-05						
4	Linear	-100074.588932	1584.31147791	52.4670636233	50.2418190274	-51.6255079463							
4	Poly	6.50742158663e-10	-3.52726638547e-05	0.0440611819696	3.50325661274e-07	-7.55507232392e-09							
3	Linear	72790.8735878	10.5112285029	0.987109870015	-34.7373243647								
3	Poly	1.47462846961e-09	-4.84255999707e-06	0.0368211442812	3.4783806537e-07								
2	Linear	6255.83497817	-110.010741204	-7.916940347									
2	Poly	6219.0872752	-109.960649299	-0.270999813849									

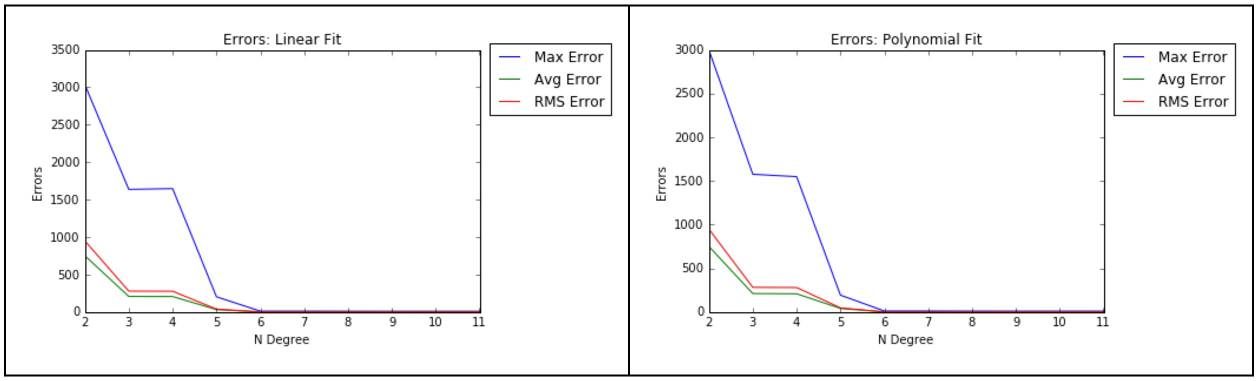
4.2 Summary of Errors

Error Table based on N Degree

N	Fit	Max Error	Avg Error	RMS Error
11	Linear	9.98979456301	0.0203793879419	0.108204039815
11	Poly	9.9921518843	0.0567686938823	0.223372888097
10	Linear	9.98734707444	0.0241456367393	0.118073644526
10	Poly	9.98844764355	0.0241489937604	0.118082808219
9	Linear	9.93534141964	0.109974969706	0.253014840851
9	Poly	9.93081352681	0.110303339363	0.253107159041
8	Linear	10.5076013001	0.580960200497	0.745496907816
8	Poly	10.4672955743	0.589075908242	0.74901334205
7	Linear	11.6349142726	1.10098287901	1.39105797219
7	Poly	11.5710958498	1.14461279068	1.42145045406
6	Linear	11.6782289316	1.10116965127	1.39170198777
6	Poly	11.6130442627	1.14741954	1.42332916407
5	Linear	202.781421152	32.0827484799	39.0501142157
5	Poly	192.873928983	40.65599374	47.2102298018
4	Linear	1647.44307487	206.405727936	277.399097597
4	Poly	1549.08368973	208.591220639	280.316678793

3	Linear	1637.14837674	207.964031037	279.044335239
3	Poly	1576.46819607	211.228884858	282.033298494
2	Linear	3032.2222257	750.248054406	948.161428499
2	Poly	2997.25513316	749.822376	947.546354333

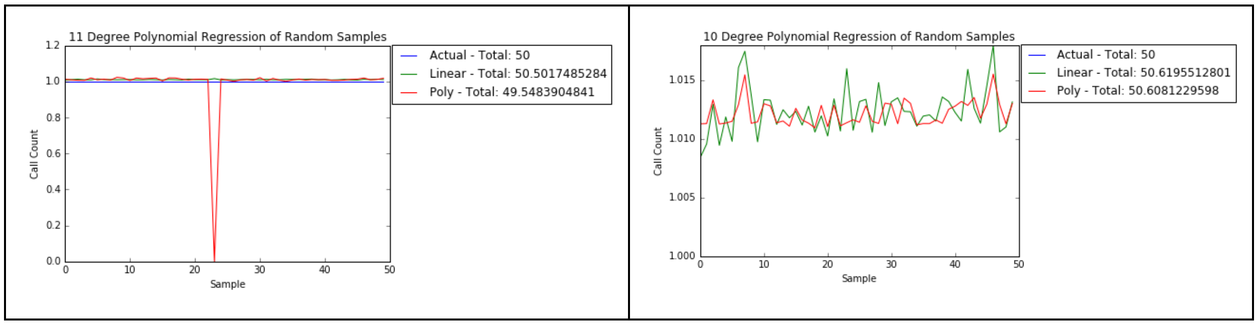
Graph of Errors for the Regression Models using Linear and Simple Polynomial Fit

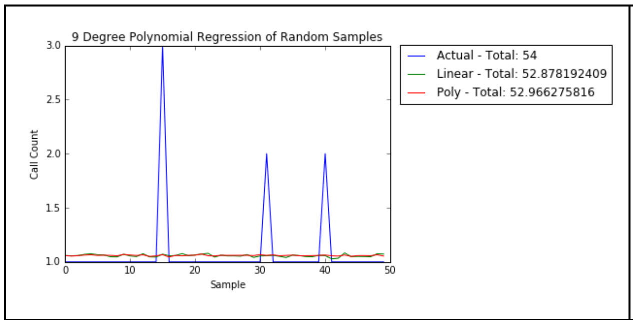


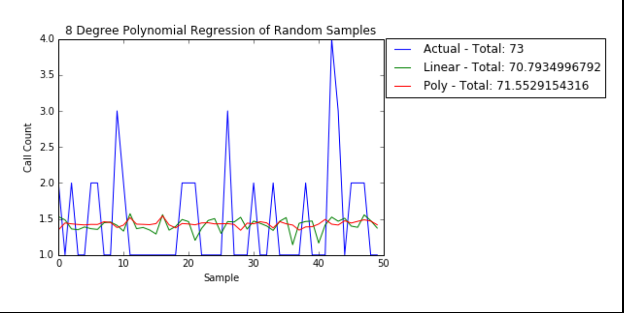
4.3 Using Train Test to Evaluate Models

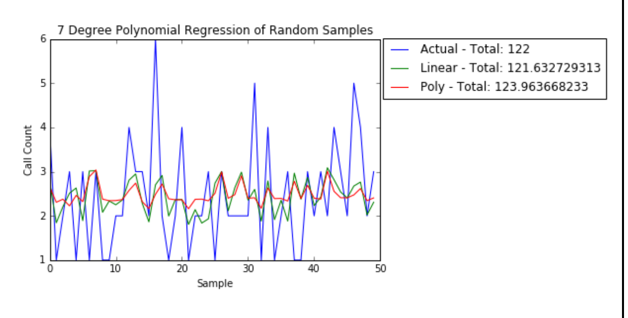
We tested each model with randomly selected samples. In each individual case, this means training or generating the curve and comparing the actual call volume to the estimation using the generated coefficients and polynomial regression.

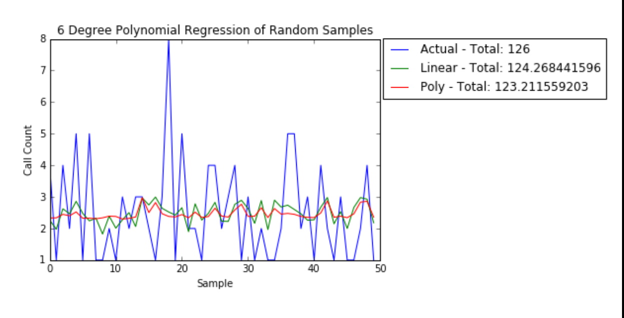
.

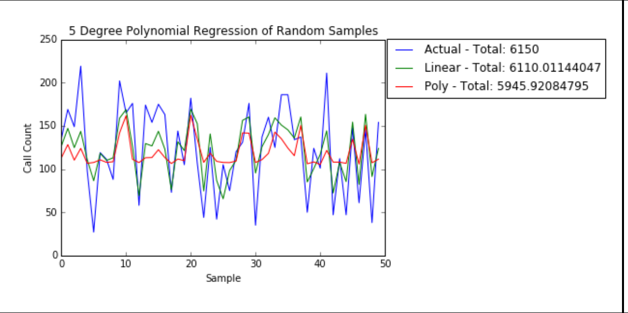


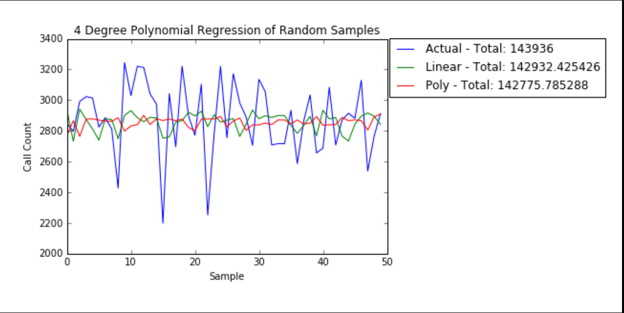


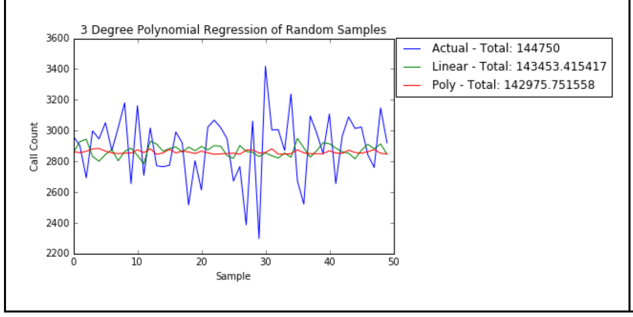


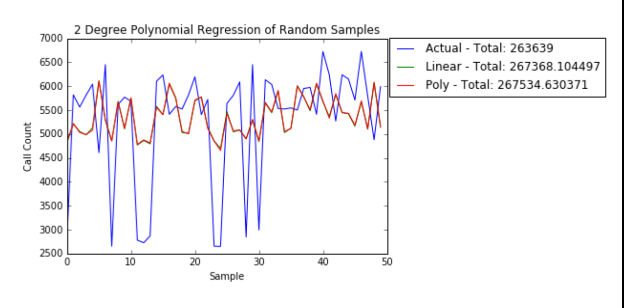












4.4 Applying the Best Curve

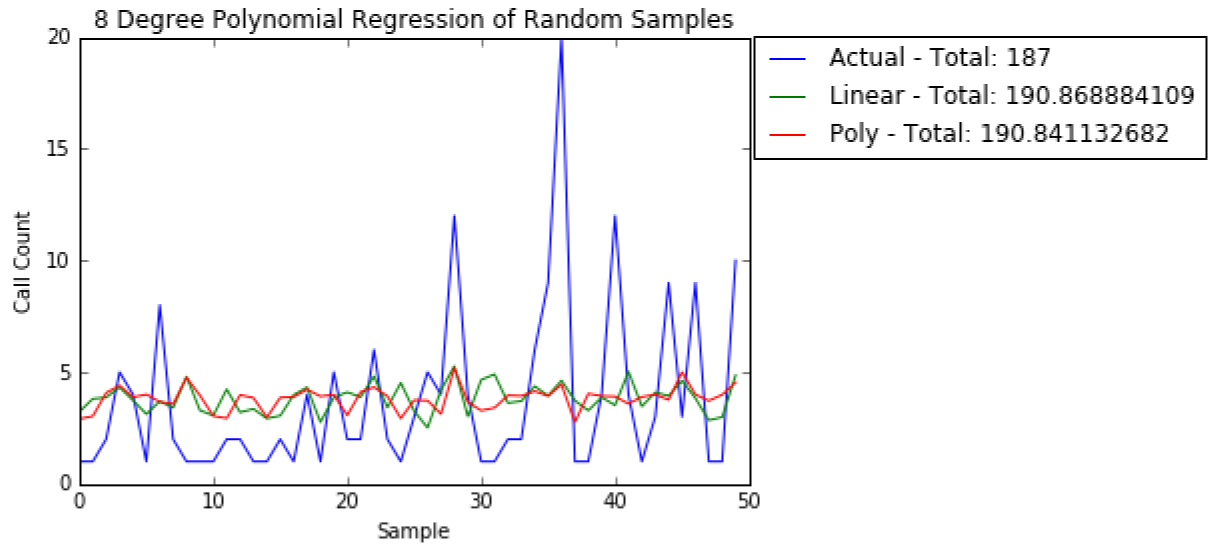
With our most effective model, we can project what the call volume will be for any given date, district, etc., and minimize the error. Based on our error table, the 11 degree model using least square fit: linear has the lowest RMS error. However, each criterion (column such as district, longitude, etc.) are not always going to be available in practical applications. There are situations when we estimate call volume based only on a certain criteria such as given date and call priority (missing district, longitude, etc.). The Python script, Main.py, creates a subset of the dataset based on user criteria, and then generates a model. The coefficients for the polynomial regression are solved using linear least square fit as it has lower RMS error (refer to the error table).

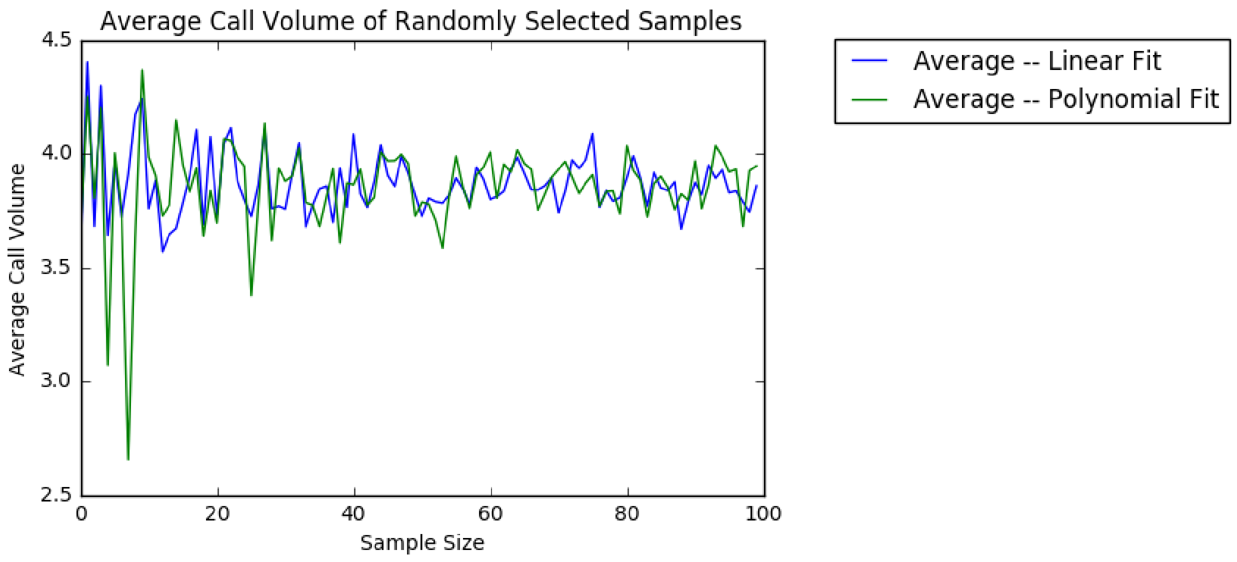
The best curve may include the following columns:
month, day, year, dayOfYear, callHour, dayOfWeek, priority, district

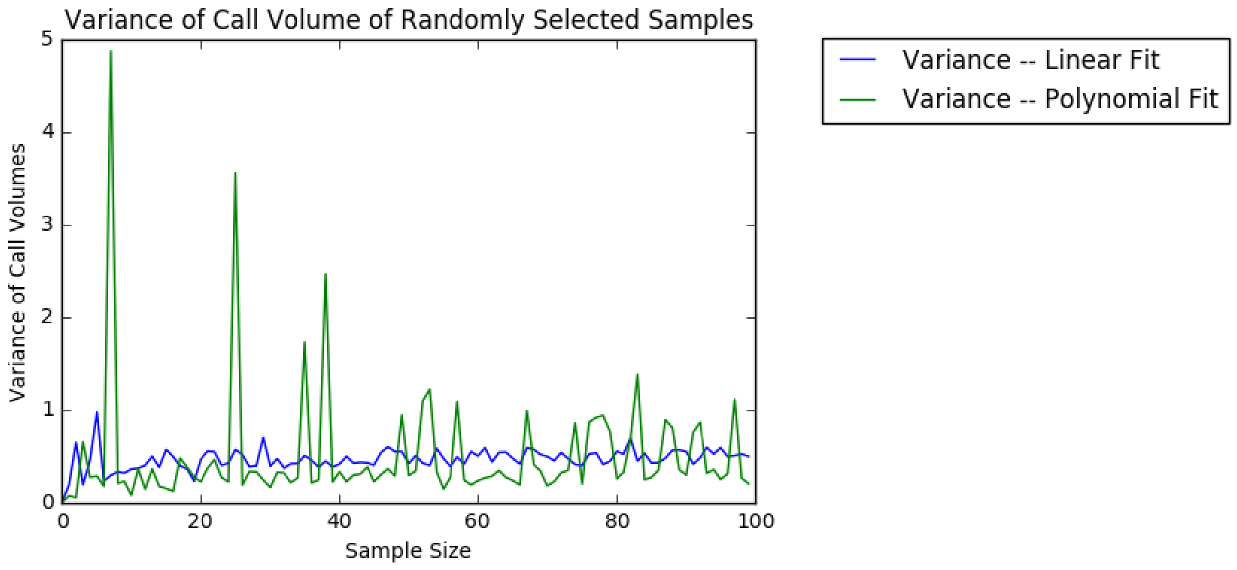
Depending on the user criteria entered, some will be omitted and a model will be generated to estimate the call volume using polynomial regression.

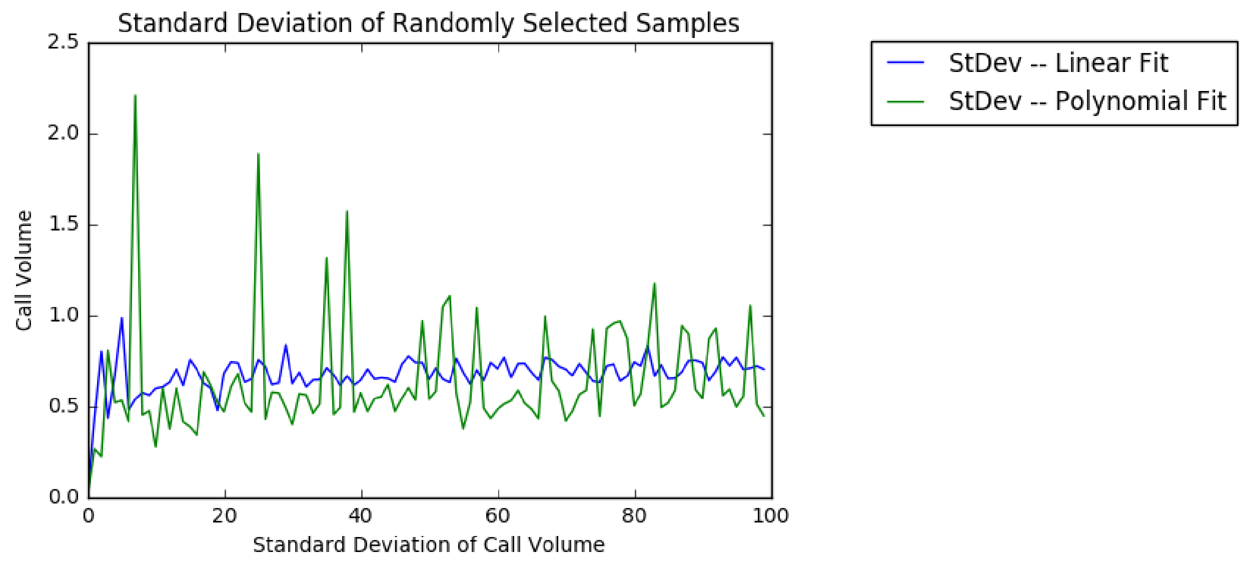
```
Least Square Fit: Linear
Polynomial coefficients:
[  3.72342215e+02  -2.35208148e+00  -7.46520812e-02  -1.82316905e-01
   7.72819301e-02   9.62887312e-02  -2.00961744e-02   2.15489426e-01
  -2.56249687e-02]
Max Error, Avg Error, RMS Error
(79.375601122473654, 2.4432059842806497, 3.3104591298100812)


Least Square Fit: Polynomial
Polynomial coefficients:
[ -1.19720170e-11  -5.25988001e-08   6.12410073e-05   4.88399025e-10
  -1.58987296e-11   1.98057759e-07  -5.98995967e-06  -4.32836190e-04
  -1.41968767e-09]
Max Error, Avg Error, RMS Error
(79.908774059443914, 2.4304570488244304, 3.3088776204900077)
```




Average Call Volume of Randomly Selected Samples
Average Call Volume
Sample Size
Average -- Linear Fit
Average -- Polynomial Fit


Variance of Call Volume of Randomly Selected Samples
Variance of Call Volumes
Sample Size
Variance -- Linear Fit
Variance -- Polynomial Fit



5 Discussion

5.1 Engineering Decisions and Consequences

We first and foremost created a program that generated a polynomial regression for a specific number of data. With the success of our preliminary testing, more columns of data were added, increasing the degree of the regression model and effectively increasing the reliability of our equation. One of the first problems we ran into in this section of the project was the massive amount of time our program spent in order to generate a regression model simply because of the sheer volume of data (over two million entries). It was excruciatingly cumbersome to test the effectiveness of our program so we took liberties to truncate the amount of data collected in order to efficiently test our program. Through this selective input of our data, we were also able to avoid any of the outlying data from events such as the Baltimore Riots which negatively impacted our results.

5.2 Strengths

The main strength comes the size of our dataset (over two million entries), and creates more data points which helps make the polynomial regression more accurate. It becomes comes more and more accurate as dataset size increases. The main driver, Main.py, allows for variable user input and estimate call volume based on given user criteria. It allows user to still get an estimate without specifying all variable inputs. However, the predicted call volume becomes much more reliable as a higher degree of polynomial regression and the errors are minimized. While sufficient data is needed in order to train the curve, once the curve is generated, most calls may be predicted with the exception of any extremities of course.

5.3 Limitations

Our best model has a few limitations in predicting the frequency of 911 emergency calls. Firstly, our model cannot take into account extraordinary circumstances such as the Baltimore riots in the spring of 2015. During this kind of event, 911 calls spike, but there is no way for the data from previous years to predict these events. This is also why we excised data obtained during the Baltimore riots in the spring of 2015 from our dataset.

The model is also limited by the data which it has access to. In other words, it would be useful to differentiate between fire, ambulance, and police 911 calls in order to examine when which specific departments are likely to experience elevated calls. Unfortunately, the dataset from the Open Baltimore site does not include that subset of data or information where it can be inferred from consistently.

The model is also limited to the independent variables specific to the ones provided in the Open Baltimore dataset. If we were to apply the model to a different state with a different set of independent variables, we would have to calculate the least square fits for the new data and create a new model based on the new coefficients. The methodology of the project, however would be identical.

6. Open Problems

Creating Visual Graphs for Functions with More than 3 Dimensions
For models that require more than 3 dimensions, it is incredibly difficult to create a visual representation for the models. As our model contained up to twelve different dimensions, we could not remedy this problem and we're only be able to provide a functional representation of the model.

7. Conclusion

Our method of generating a regression equation and model in order to predict the volume of 911 calls was indeed successful. With Open Baltimore providing such a wealth of data for us to analyze, there is no shortage of data needed in order to create our desired equation.

Utilizing python, we were able to download, parse, and clean as much data as we needed. We then used polynomial regression and least square fitting in order to organize all of the data into a model that we could use to predict the volume of calls. An existing library was used in order to perform Gaussian Elimination which figured out all of the relevant coefficients of the regression equation. Once the backbones of the program was complete, we cleaned up the program, made it more efficient for testing and reproduction, and made the data processing much more selective.

Once the regression equation was formed, we could clearly see that the equations that utilized more parameters of data (higher degree) produced far less errors which made it much more accurate at predicting the amount of calls. While any extremities remained unpredictable as expected, we could see that the polynomial curve was very much effective at predicting what we needed to know.

In the future, we hope to use our results to predict the call volumes for days, maybe even months, ahead of time while also predicting the calls down to a specific time, before the calls are even transmitted and recorded onto Open Baltimore. In this way, we can truly see how effective our regression model will be.

References

Calls for Service from Open Baltimore. Retrieved October 08, 2016, from https://data.baltimorecity.gov/Public-Safety/Calls-for-Service/xviu-ezkt